Copytrading strategy analysis in ETHEREUM using ML

Jaime Fábregas Fernández, *Tarlogic Security*.

***Abstract*—This document provides insights about the copytrading strategy within the Ethereum blockchain ecosystem. This ecosystem has experienced an explosion in recent years in terms of growth and provision of applications and services, including financial services. These are the so-called DeFi (Decentralized Finances). Among these services are decentralized exchanges, which allow cryptocurrency trading. These services are carried out through *smart-contracts*. The addition of *smart-contracts*, i.e. executable programs within the blockchain, has been Ethereum's great contribution to the world of cryptocurrencies. Uniswap is one of the most widely used and popular *smart-contract* for cryptocurrency trading. That is why it is the one selected for the analysis.**

This document will detail the methodology, the designs of the experiments to test the hypotheses, and the final results.

The experiments have been structured around the use of Machine Learning techniques that allow the prediction and/or ranking of the best trades to be replicate from successful traders, measured in terms of net profit.

The final results expose serious doubts about the feasibility of this technique in a deeply random environment such as crypto-trading. Despite having detected weak relations that have helped to make better estimates, given the hostile environment and the high probability of incurring in losses, these positive relations are not sufficient to overcome this downward trend and be able to make profits.

I. INTRODUCTION

Ethereum's blockchain has enabled the creation of an ecosystem consisting of decentralized applications that can replicate real-world financial entities. This is the case for banks, games, but especially financial products such as cryptocurrency exchanges. We will focus on the latter.

A. Copy trading

This technique is based on the identification of successful traders. Once identified, their transactions are copied to obtain similar returns.

The starting hypothesis is as follows: those individuals with successful trades are more likely to be successful in their future trades. This may be either because they have access to better sources of information or because they have developed effective techniques, methods or even intuitions. In the development of this analysis, the effectiveness of the "*copy trading*" strategy will be studied.

II. DATASET

In this analysis, all Uniswap V2 and V3 transactions on the Ethereum blockchain were processed. These transactions started in May 2020 and were processed up to March 2022. In total, 43 million transactions.

Each transaction contains the following information:

- Wallet (whether it is a person/bot/platform).
- Tokens being traded.
- Amount exchanged of each token.
- Block and date of transaction.
- DEX (V2 or V3)

III. DATASET PREPARATION

Before carrying out any statistical analysis it is necessary to prepare the data. We divide it in two steps: collection and filtering.

A. Data collection

All transactions are processed and grouped by wallet. A list of all wallets that made transactions is obtained, and the data is aggregated for each one of the registers. Additionally, they are grouped by month, so that the same wallet can only appear once per month.

Each row represents a buy transaction and incorporates additional information about the originating wallet, pricing analytics of the token being purchased, and the final result of the transaction measured in profit percentage.

The columns included in the dataset are as follows::

- Wallet
- Previous success rate of the wallet.
- Activity rate.
- Average profit grouped by token.
- Median profit of all transactions.
- Median amount of ETH used in transactions

This analysis was carried out by Tarlogic Security, one of the cybersecurity and cyberintelligence companies with the fastest growth in Europe. It has also developed the communication analysis software Acrylic WiFi.

- Token age (in logarithmic scale of hours)
- Token price variance in the last 24H
- Token price variance in the last week
- Token price variance in the last month
- Token volume variance in the last 24H
- Token volume variance in the last week
- Token volume variance in the last month
- Profit from the fourth last transaction
- Profit of the third last transaction
- Profit from the penultimate transaction
- Profit from the last transaction
- Amount in ETH of the current transaction
- Date
- **ResponseBenefit**: Expected profit
- **ResponseMaxBenefit**: Max Expected profit

The dataset contains an entry for each transaction, along with the additional data from the originating wallet and its result (profit).

Both response variables are defined below:

- **ResponseBenefit**: Profit obtained from this transaction at the time of selling. If no sale occurs during the next 30 days, the profit will be calculated based on the token price 30 days after the purchase, as if it was sold.
  It is necessary to define a limit until we will wait for a sale to happen. In this analysis we have defined that limit to 30 days.

- **ResponseMaxBenefit**: This is the maximum profit that could have been made on this transaction by selling at the highest price during the next 30 days after the purchase. The price at which it is sold is at the 95th percentile. Thus, this measure of response can be considered *slightly conservative*.

  The rationale for choosing the 95th percentile is to avoid outliers, reducing abrupt price spikes and drops that would introduce significant inaccuracies into the data set. Using percentiles provides a more robust measure.

B. Data filtering

The next stage in data preparation is filtering. Again, in order to proceed with statistical analysis it is important to eliminate outliers that may compromise the analyses.

In this analysis no statistical method for outlier detection was used. It is however, advisable to use these methods for future research.

To reduce the noise in the data, it was decided to start by filtering out those wallets that have less than 5 transactions. This is, intuitively, important to eliminate those wallets that have hardly any transaction history and therefore it would not make sense to copy their operations, since there is not enough data to assess whether they are successful or not.

The study population is thus reduced to those wallets that have interacted with DEX at least minimally, according to the defined threshold. Limiting the data to the population subset of interest allows ML algorithms to refine the result, narrowing the search space.

The total number of transactions after the first filter is: 96,783.

Next, the maximum and minimum values are established for each variable, eliminating those rows that do not fall within the defined ranges:

TABLE I
APPLIED FILTERS

Filter	Eliminated	Percentage
Transaction ETH < 50	963	0.99%
Profit < 15	252	0.26%
Median ETH < 40	268	0.27%
-100 < Variances < 100	991	1.02%
ResponseMaxBenefit < 50	71	0.07%
	Total	**2.54%**

IV. TEST AND TRAINING DATASETS

The dataset is divided into training and test datasets, necessary for any analysis with ML involved..

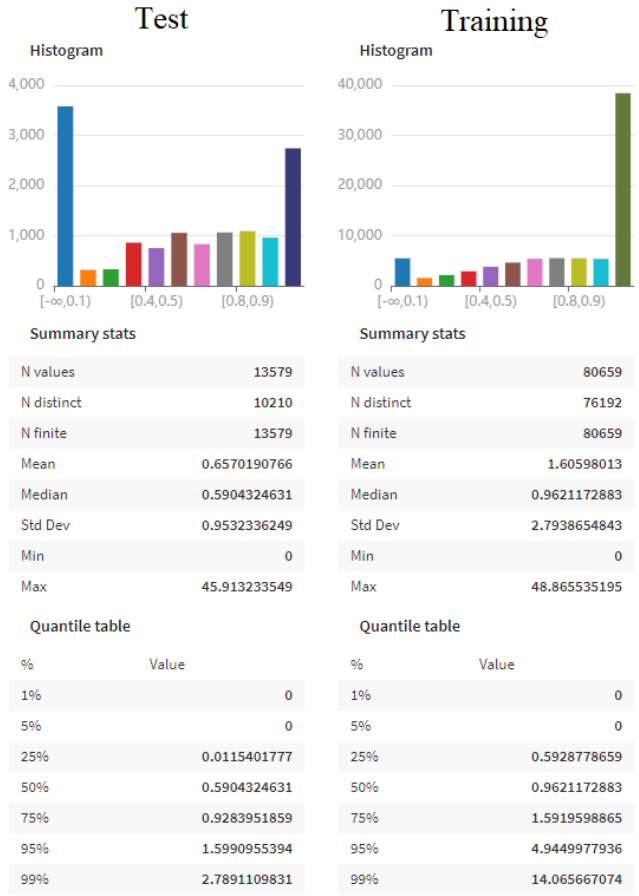

Fig. 1. ResponseMax feature analysis on *test* and *training*

The nature of the data is strongly dependant on the time variable, so it is important to split the data sorted by date. The training data corresponds to 85% of the oldest registers, whereas the test data to the 15% most recent.

This time-dependency manifests in significant differences between statistical parameters, as shown in **Figure 1**.

It is important to note the differences between the two data sets, in particular, the very disparate proportion of transactions that have a profit of 0 between the two data sets (meaning a 100% loss).

Figure 2 shows the large difference between profitable and non-profitable transactions between the test and training datasets (blue being non-profitable transactions).

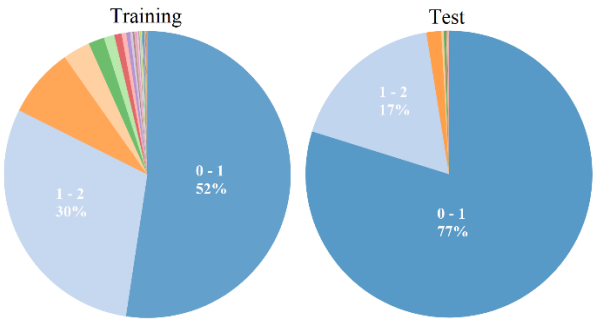

Fig. 2. Pie chart for feature **ResponseMaxBenefit**

A. Implications on the ML results

The data show a very important volatility and a strong time dependency. This makes it even more challenging to obtain ML predictions. Almost half of the transactions in the training dataset are profitable, while less than 20% are in the test dataset.

This implies that an improvement sufficient to produce a benefit in the training set may not produce a benefit in the test set. The improvement needs to be *very significant* to produce a benefit on both sets.

B. Profit calculation problem

Expected profit calculations on the data sets are not trivial, since they depend not only on **ResponseMaxBenefit**, but also on the *target* defined, which is arbitrary. If the *target* is not reached, the token will not be sold. In this case, the final profit will be set based on the token price 30 days after the purchase, when the sale would necessarily take place (because of previous requirements). If that happens, the profit would fall to what is shown in the **ResponseBenefit** variable.

IV. DATA ANALYSIS

After the elimination of outliers, an analysis of the data is performed to detect anomalies. Two types of analysis are performed: univariate and multivariate.

A.Univariate analysis

An univariate analysis is performed for each variable, including predictors and response variables. The distribution of all variables is included below in order from left to right and from top to bottom. The date variable is excluded because it is linearly increasing and has no outliers.

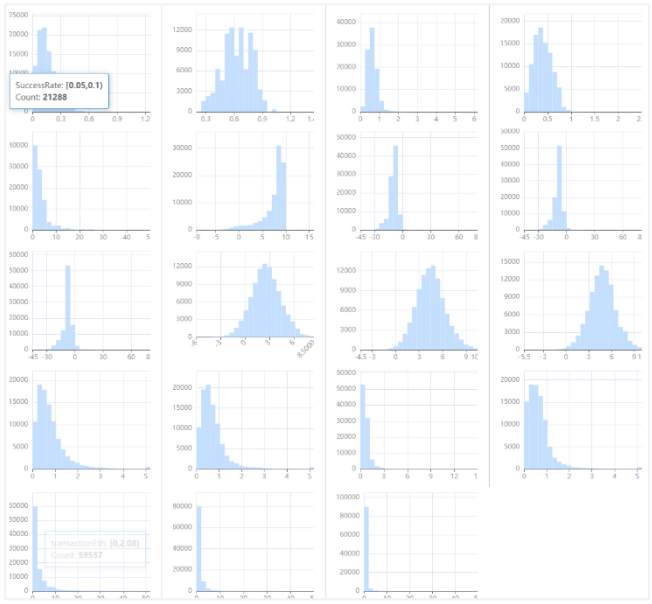

Fig 3. Distribution of 19 variables after filters were applied

The most extreme distributions correspond to the two response variables (bottom right). It was decided not to further narrow these two distributions because they are already limited to a maximum profit of 50 times the initial investment.

Although this extreme value is rare, it is perfectly possible for it to occur within a month. This is true specially among young tokens, without this necessarily being an outlier. Quite the contrary, we would like to find some relations between these expected returns and the predictors.

B. Multivariate analysis. Correlation.

To perform a first analysis of the relations that may exist between predictor variables, a correlation heatmap is carried out. The coefficient chosen is Spearman, in favor of Pearson's, as it is better at discovering monotonic nonlinear relations between variables.

This heatmap will also help to discover multicollinearity problems, if any. Additionally, the predictor variables will be compared with the response variables to discover possible strong relations between them that may be of great interest.

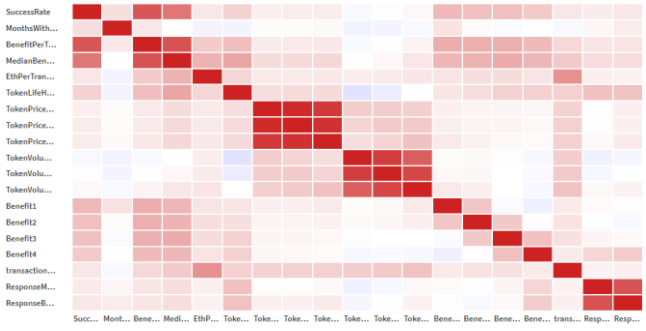

Fig. 4. Spearman correlation heatmap

Both response variables (last two in the table) do not show a strong or moderate correlation with any other variable (maximum: 0.27).

Additionally, a strong correlation is observed between the groups of variables measuring the variance of volume and price respectively (maximum: 0.96). This makes sense since changes in the last 24 hours necessarily affect changes in the last week and month.

Since these effects are limited to these two groups of variables and it may be useful to have variables representing a minimum of three time periods, i.e. day, week and month, we decided to keep these variables.

Additionally, there is also a correlation (0.76) between the profit per token and the median profit with the success rate. This is also within expectations, since a higher success rate generally increases the median profit percentage.

C. Multivariate analysis. PCA.

To assess the possibility of reducing the number of predictor variables, a PCA (*Principal Components Analysis*) dimensionality reduction algorithm is applied. The rationale behind this decision is to find out if most of the variance can be explained with a reduced number of variables.

In addition, this method can also resolve possible cases of multicollinearity present in the data set.

The result of the PCA over 17 predictor variables (response variables are excluded) is as follows:

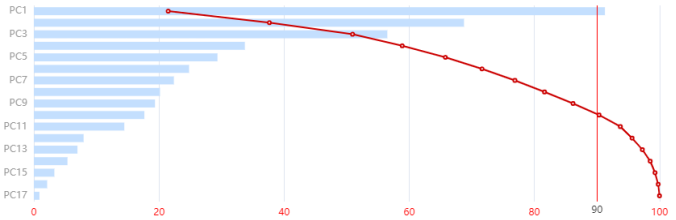

Fig. 5. Explained variance ratio

The first 10 components explain at least 90% of the variance, as shown in **Figure 5**.

By looking at the PCA heatmap, as shown in **Figure 6**, we can get a better understanding of what is being captured in the different components.

Knowing that the first 10 components explain 90% of the variance, we can focus on the last ones. Specifically, the last 4 components (PC14 - PC17) are concentrated precisely on the variables that measure the variance of Price and Volume, having almost null values for all the other predictor variables.

This makes sense since a strong correlation between them had already been detected prior to the PCA. Therefore, the relations between the las 4 components (PC14 - PC17) provide very little information, and consequently, can explain very little of the remaining variance.

In summary, the PCA reduces the number of predictor variables from 17 to 10 with a threshold of 90% of explained variance. This reduction is mostly caused by the reduction of the weights in the price and volume variance variables (last 4 components). This factor was already taken into account in the previous correlation analysis and it was concluded that they should be maintained.

In conclusion, the PCA does not provide a significant advantage in the treatment of the data that could lead to a difference in the results of ML algorithms, since the reduction in dimensionality is very moderate, and increases, on the other hand, the complexity when interpreting each of the components.

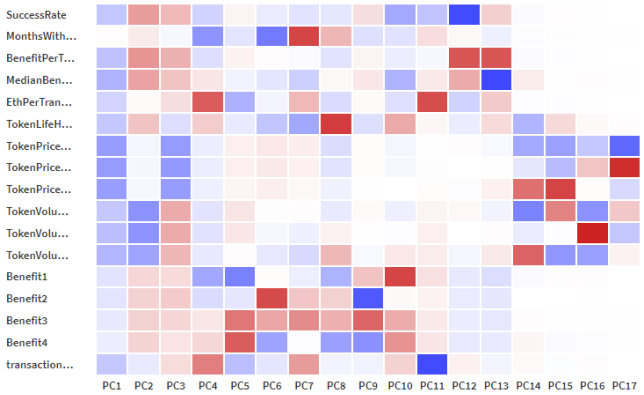

Fig. 6. PCA Heatmap correlation

For all these reasons, the PCA components will not be used in the ML algorithms that will be used later.

In addition, it may be interesting to note that tests were also carried out using the same approach designed in this project with the results obtained from the PCA, instead of using the raw data directly. This approach did not offer any significant advantage.

V. Clustering

The next step is to subject the data to clustering techniques. The objective is to identify those unsupervised learning algorithms that can extract knowledge from the data based on their grouping into clusters.

In this case, the goal is not to extract a certain number of known classes from the data, but to allow the algorithm to group the data in order to later compare the different clusters based on their distribution of the response variables.

The method used for cluster scoring is the *silhouette coefficient*, using a Euclidean distance metric. It measures how far apart the different clusters are from each other. However, although this is an important measure, in our particular case, we are more interested in measuring how efficiently the clusters separate high-benefit transactions from low-benefit transactions.

The intuition behind this is that there should be **groups that perform better or worse based on the predictor variables**. It is important to note that the response variables were not used for clustering, since this would have invalidated the groups generated by having access to future information (profit to be obtained) from each transaction.

We study this way whether the information extracted by the clustering algorithm can be valuable for machine learning algorithms and its capacity to help extract information by first categorizing it.

Five of the most popular clustering algorithms have been evaluated. The results are included in Table II.

TABLE II
CLUSTERING ALGORITHMS

Algorithm	N Clusters	Scoring
K Means	4	0.17
Gaussian Mixture	5	0.11
Mini-batch KMeans	3	0.17
Interactive clustering	5	0.19

Different combinations of parameters have been tested for each of the algorithms. The Scoring field reflects the highest score obtained for each type.

All had very similar results. Identical clusters are found in all of them, with clear tendencies to value the same set of variables, despite differing in number.

We next study how clustering affects the composition of the response variables. The same pattern is observed regardless of the algorithm chosen and the parameters set. The result of *KMeans* with 4 clusters and one additional outlier cluster is included below.

A mosaic plot of the clusters is made with the response variable stacked on each bar in one-unit intervals.

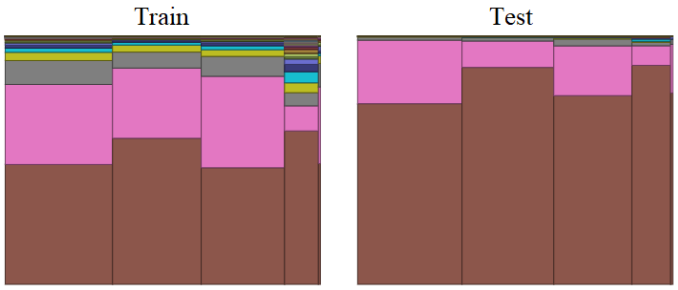

Fig. 7. Cluster mosaic plot with response variable

Figure 7 shows the result of applying *KMeans* on the training data and on the test data. When applying it on the test data set, the algorithm was not re-trained. The same centroids obtained during training have been used.

Each vertical bar represents a cluster. Within each bar, the values of the response variable showing the benefit (**ResponseMaxBenefit**) are stacked in intervals of one unit. Thus, the brown region represents the number of elements with a profit in the interval [0,1) (0 represents a 100% loss and 1 the permanence of the initial amount, with no profit). The pink region represents the elements with a response value in the interval [1,2), and so on.

The different proportion of the brown loss region is due to a time dependency effect, since the test data is obtained taking into account the date of the items and contains the most recent items. In that subpopulation the proportion of transactions with losses increased significantly.

It is shown in **Figure 7** that clusters can separate groups with higher and lower proportion of profitable transactions (regions above the brown bar). This occurs consistently in both the train and test datasets.

Because of this, we decided to include information of the cluster in each element of the dataset so that it could be used by different ML algorithms.

VI. Machine Learning. Classification.

The first approach is to use the data for a prediction by classification of two classes: profit or loss.

For this purpose, the data set is prepared so that the response variable will only have two values, representing the two classes: 0 or 1.

A. Data filtering by cluster

Two training datasets will be used:

- **Full:** with cluster information included.
- **Filtered** by cluster 0. (Increases the proportion of transactions with profit, at the cost of reducing the total number of records to 21,331).

The intuition behind this decision is to compare the accuracy of the algorithms when they have cluster information or when they have a data set more favorable towards positive transactions thanks to the filtering of one of the clusters.

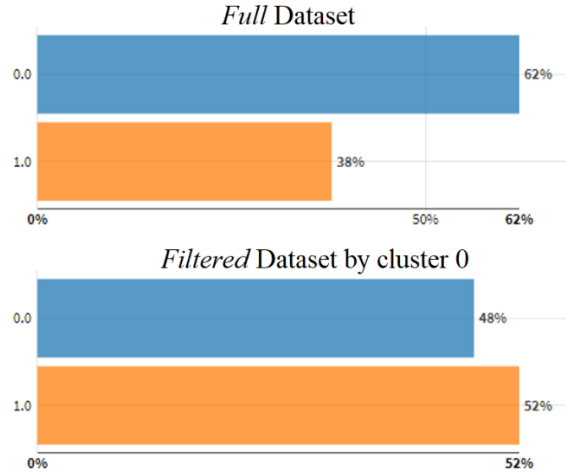

Fig. 8. Ratio of classes in the two training datasets

As shown in **Figure 8**, the application of the cluster on the data results in a significant increase of positive elements (class 1).

B. Metrics.

ML algorithms have been parameterized to optimize the **accuracy** metric. This is because the main interest is to *reduce the number of false positives*.

Once in a production environment, only transactions with positive prediction will be executed, while no transactions with a negative prediction will be copied (using the *copytrading* strategy).

False negatives do not cause a financial loss in this strategy, but there is such loss in the case of false positives, if a transaction that is going to produce a loss is copied. This is why the main interest of predictions is to reduce false positives.

Only once the proportion of false positives is sufficiently low, it would be reasonable start working on reducing false

negatives in order to increase the total number of transactions copied, thus increasing the profit in absolute terms.

C. Cross-validation

The well-known *K-fold cross-validation* method is used to obtain a precise measure of the accuracy of the algorithms. In this method, the total set of training data is subdivided into an arbitrary number (commonly 5, in our case). Four portions of the data are assigned to training whereas one to testing, permutating through all 5 possible combinations.

In our particular case, since it is necessary to maintain the data ordered by date, this is done in a similar way, as shown graphically in **Figure 9**.

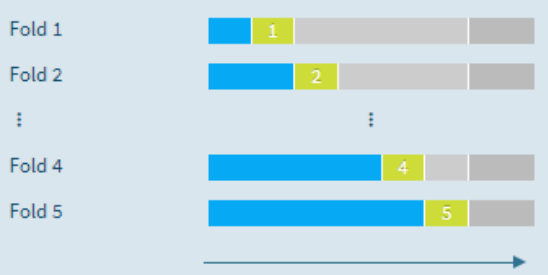

Fig 9. Cross-validation with data sorted by date

D. Test data preparation

The training data it is again separated into a test and training set. In total we have two test sets: one global and another that will be used to measure the *accuracy* of the different ML algorithms. When using several ML algorithms on the same test set, the probability that at least one of them will give a positive result by chance increases. Therefore, a second test set is reserved to test only the ML algorithm that is selected as the best and returned a positive result. This is due to the nature of the present analysis where it is critical to reduce losses.

When creating the test data, the date is still considered, so that the data will be sorted chronologically. All items in the test set will have a higher date than any item in the training set.

E. ML Algorithms selection

First, four ML algorithms were selected with the goal of maximizing their differences between the approach to find a solution. These are:

- **Logistic Regressio**n. One of the best known and simplest. It is selected to act as a baseline to compare other algorithms.

- **Random Forest**. It is an algorithm that involves a change in the paradigm of solution search through decision trees. That is, it is different in nature from logistic regression, and this is the point we are interested in evaluating.

- **SVM**. Could not be missed. It has exploded in popularity in recent years. It is a method based on folding hyperspatial curves (manifold) to classify. Again, in this case it is of interest that its nature is completely different from the previous ones.

- **Neural network.** In this case a classical multilayer perceptron neural network is chosen, with three layers, *tanh* as the activation function and ADAM as the optimizer. It is not intended to develop an optimized architecture to find a particular pattern, since at this point we have not yet developed any hypothesis in this regard. That is why a classical architecture is chosen to measure its performance.

F. Results

The results are shown in Table III below.

TABLE III
F1 SCORING BY DATASETS

Algorithm	Full	Filtered
Logistic Regression	0.423	0.495
Random Forest	0.469	0.502
SVM	0.433	0.492
Neural Network	0.429	0.492

Table III shows the F1 scoring. The results are slightly better than a random classifier. In addition, it should be noted that the accuracy measures do not differ significantly.

It is important to study the decision graph to discover the behavior of the algorithm based on the threshold set to separate classes.

A classic threshold is 0.5, however, it is common to adjust these thresholds to increase the *accuracy* so that we allow the algorithm to give an opinion only in those cases where it is "very sure" of the result.

The decision chart of the Random Forest algorithm for the *Complete* dataset is included below, where it can be seen that there are no significant differences in the variation of the threshold, except in extreme cases where the cases are reduced to one or none.

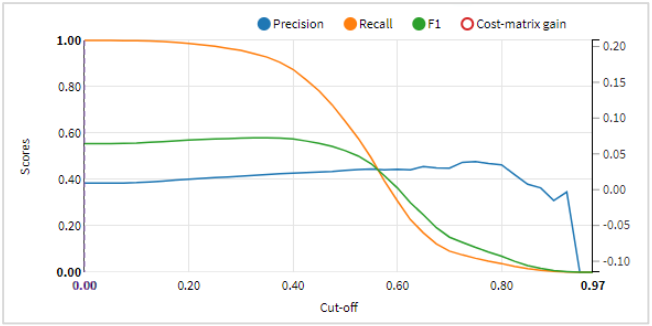

Fig. 10. Random Forest decision graph in *Full* dataset

Figure 10 shows that the *accuracy* hardly varies with increasing threshold (blue line). This effect occurs analogously in the rest of the algorithms and datasets. Not all of them are included for convenience.

Regarding the results per dataset:

- **Full**: a *slight improvement* from 0.38 of positive cases (**figure 8**) to 0.469 in the best case. The gain chart (**Figure 11**) is included below, where the small gain corresponding to the Random Forest algorithm can be appreciated.
  The area under the curve (AUC) both by quantity and shape is far from the desired shape (orange line at best).
  The cumulative gain (blue line) has a slope very similar to that of the random model, and specially in the initial values of the x-axis, which is where it is of most interest, indicating almost zero performance.

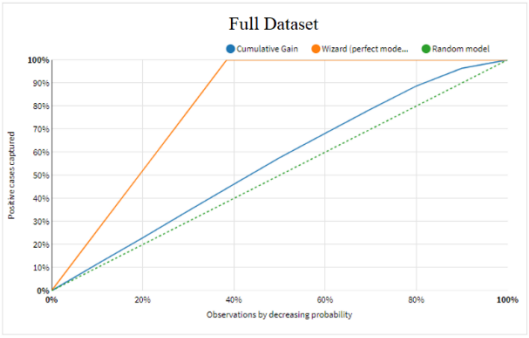

Fig. 11. Random Forest Gain chart in *Full dataset*

- **Filtered:** we observe that the algorithms have an accuracy similar to a random classifier that takes into account the mean of positive cases in the population: 0.52 (**Figure 8**). Therefore, *there is no significant improvement*.

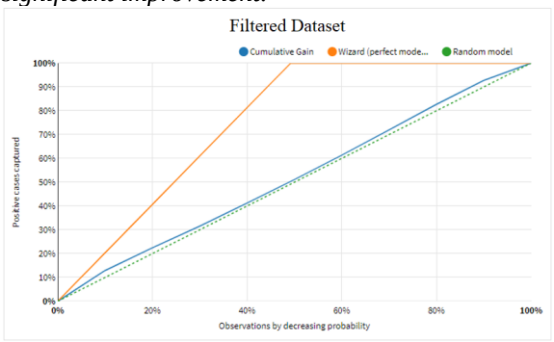

Fig.12. Random Forest Gain chart in *Filtered*

It is remarkable how both Random Forest gain charts, **Figure 11** and **Figure 12,** hardly offer any improvement (same happens with the rest of the ML algorithms, Random Forest was chosen here as a representative example). More specifically, the one corresponding to the *Filtered* Dataset, hardly differs from the base diagonal line corresponding to a random model.

G. Conclusion

The ML algorithms studied failed to increase the success rate in the case of data already filtered by clustering and do not reach the success rate obtained by clustering when not filtered.

We can therefore conclude that:

1. Clustering (unsupervised) is more efficient than supervised ML algorithms in separating positive and negative cases.

2. Supervised ML algorithms are not able to find any relation that clustering did not already find.

The maximum expected improvement is **0.502**. This accuracy is insufficient to make a profit, since most of the transactions with loss, will lose everything, whereas profitable transactions rarely duplicate.

In other words, a much higher accuracy is needed to reach the profit zone.

We conclude that this approach is not profitable in a copytrading strategy.

VII. Machine Learning. Prediction

A new approach is defined where the algorithms will be used not to classify but to perform the prediction of the expected benefit.

In this way, more information is incorporated into the ML algorithm, since it is now a continuous variable instead of a binary one, so that, intuitively, the algorithms can know not only whether they were wrong or right, but also by *how much*.

A. Metrics

There is a major problem when choosing a metric in ML prediction algorithms in projects of this type. This is because a classical metric such as RSME can provide an idea of how far the prediction is from the real value, but it does not provide information on whether the prediction was correct or not.

In our case, we are interested in knowing whether or not a transaction will be profitable in the first place. That is, if the prediction is greater or less than one. An RSME value does not provide relevant information since the algorithm may be getting right if it is a profitable or loss-making transaction, but not by how much and vice versa.

Therefore, the two metrics that are implemented are:

- **Spearman correlation**. Spearman is chosen in favor of Pearson for its better reflection of monotonic nonlinear correlations. In addition, it also represents linear relationships well.
  The intuition behind this choice is that if there is a correlation between the prediction and the response variable, it is possible to adjust the threshold to increase the proportion of true positives (*accuracy*) until benefit is achieved.

- **Real expected profit**. This is a custom metric for this case. The real expected benefit is calculated based on the response variables. Here are the steps:

  1. If the prediction is less than one, it is ignored.

2. If the prediction is greater than one, the target is set as the prediction.
3. If the target is reached, then the profit is equal to the target.
4. If the target is not reached, then the profit is equal to the *ResponseBenefit*, which is the profit after copying the sale or making the sale after 30 days if the sale has not occurred.
5. The total result is divided by the number of transactions processed (those with prediction greater and less than one).

- **Real expected profit with fixed target.** It is the same metric as the previous one but a fixed target is established instead of a dynamic one based on the prediction.

B. Cross-validation

This section is analogous to the same subsection in VI.

C. Test data preparation

This section is analogous to the same subsection in VI.

D. ML Algorithms selection

For the metric "*Real expected profit with fixed target*", only the two most representative ML algorithms were selected: Random Forest and Neural network.

The reason is to avoid overextending the tests, since they will be repeated for different target values. Only in case of obtaining results close to 1 (without loss) we would contemplate testing the rest of the algorithms and starting a process of hyperparameter refinement.

For the other two metrics a wide range of ML algorithms will be used, as a last option before totally rejecting the copytrading strategy. These are:

- Random Forest
- Gradient Boosted Trees
- Lasso
- Light GBM
- XGBoost
- Decision Tree
- SVM
- SGD
- KNN
- Extra tres
- Lasso-Lars

E. Results

The results with the first two metrics are first included below, in Table IV.

It is important to note that both scores are completely different in nature, however, they should tend towards one.

In the *Spearman* case, values close to 0.6, which are common in the *Real* scoring, would generally imply benefits higher than 1. The scoring that best reflects the benefit obtained is the *Real*, since it directly measures the benefit, while in the *Spearman* case, it has to be measured at a later stage.

TABLE IV
MODEL SCORING BY METRIC

Algorithm	Spearman	Real
Random Forest	0.025	0.590
Gradient Boosted Trees	-0.065	0.576
Lasso	-0.143	0.632
Light GBM	-0.008	0.638
XGBoost	-0.069	0.607
Decision Tree	0.024	0.625
SVM	0.199	0.386
SGD	-0.106	0.607
KNN	0.050	0.468
Extra trees	-0.051	0.617
Lasso-Lars	-0.146	0.613

A. Spearman

Starting with the Spearman metric, Table IV shows results very close to zero, indicating that it has not been possible to establish a satisfactory relation between response variable and prediction. Not even a moderate one.

The highest score corresponds to the SVM with 0.199; however, after testing this model with the second set of tests (more recent), the Spearman correlation is again very close to zero.

Two algorithms (SVM and Random Forest) are evaluated with the second test set. This assumes a very accurate evaluation by eliminating the bias of choosing the best algorithm. Since we are repeating the experiment with 11 different algorithms, this could be considered a type of *p-hacking* by increasing the possibility of a false positive. To eliminate this effect, we double-check the result with a different and more recent test set.

Since Spearman scoring refers to a correlation, it implies that increasing values of the prediction should be related to increasing values of the response variable. However, the prediction values do not necessarily provide information about the success or loss of the transaction, since they are a relative measure. It is important to emphasize that this is because the metric used is a correlation that does not inform the ML algorithm about what the actual profit value should be. It only tries to maintain the increasing relation between predictions and responses.

Therefore, when measuring the actual accuracy in the test set, no thresholds are set on the prediction variable. Instead, predictions are ordered and percentiles are used to measure their *accuracy*.

TABLE V
RF MODEL SCORING IN TEST DATASET

Percentile	Profit
99.7	0.57
99	0.46
90	0.19

Table V shows negative results, with very significant losses. This option is rejected.

B. Real

In the actual profit measurement, many of them have values around 0.6 profit.

In the test set used, it is possible to obtain this result by simply accepting all transactions and setting a target of 1.2.

This is why, again, the algorithms show that they are not able to find patterns that outperform a random model using the average of profitable transactions.

This is proven by subjecting them to different test sets and checking that the performance varies in the same proportion as the number of positive transactions in the datasets.

Therefore, given how far away the scoring is from the profit zone and the indications that the algorithms are not being able to capture relevant information from the data set, this option is also rejected.

C. Real with target

In this case, the algorithms are limited to two, in order to reduce the total number of tests. Their behavior is studied with different target thresholds (threshold at which the sale is made).

TABLE VI
MODEL SCORING WITH FIXED TARGET

Target	Random Forest	Neural Network
1.2	0.634	0.611
1.8	0.674	0.649
2.2	0.687	0.661
2.6	0.693	0.668
3	0.702	0.676
4	0.708	0.683
6	0.707	0.687

The result obtained is very similar to the *Real* metric. In this case, however, the best of the cases overcome the 0.7 barrier. However, it should be noted again that these results are virtually identical to running all transactions with a high target. It is therefore related to the nature of the data rather than the ability of the algorithm to capture information from it.

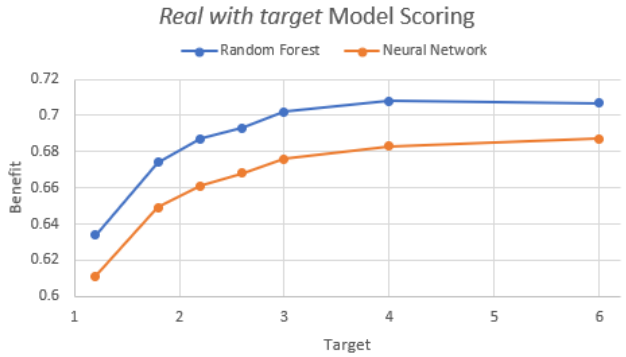

Fig. 12. Profit per fixed target

Similarly, when tested with the most recent test set, this benefit plummets to 0.3, due to the different proportion of positive transactions.

Therefore, this option is also rejected.

VIII. Conclusions

The *copytrading* strategy is not feasible. With the ML methods known so far and using the techniques studied, it is not possible to make a profit with this technique by a wide margin.